Remarks

This REPLY is in response to the Office Action mailed September 16, 2008.

I. Summary of Examiner's Rejections

Prior to the Office Action mailed September 16, 2008, Claims 1-7, 9, 11-17, 19, 31-34 and 37-42 were pending in the Application. In the Office Action, all of the claims were rejected under 35 U.S.C. 103(a) as being unpatentable over Taylor et al. (U.S. Publication No. 2004/0019897, hereafter Taylor) in view of Susarla et al. (U.S. Patent No. 6,915,511, hereafter Susarla).

II. Summary of Applicant's Amendments

The present Reply amends Claims 1 and 11; cancels Claims 5, 15 and 37-42; and adds new Claims 43-46, leaving for the Examiner's present consideration Claims 1-4, 6-7, 9, 11-14, 16-17, 19, 31-34 and 43-46.

III. Claim Rejections under 35 U.S.C. §103

In the Office Action, Claims 1-7, 9, 11-17, 19, 31-34 and 37-42 were rejected under 35 U.S.C. 103(a) as being unpatentable over Taylor (U.S. Publication No. 2004/0019897) in view of Susarla (U.S. Patent No. 6,915,511).

Claim 1

Applicant thanks the Examiner for the detailed "Response to Arguments" provided by the Examiner in the Office Action mailed September 16, 2008. Accordingly, Claim 1 has been amended to more clearly define the embodiment therein as:

> *1. A system for loading software applications, comprising:*
>
> *a server, executing a Java virtual machine that includes a system classloader, for storing and running a plurality of software applications, wherein each of said plurality of software applications includes a plurality of deployable modules and classes associated therewith, and wherein the software applications can be customized by a software developer and then deployed to run on the same server;*

a control file, that can be edited by the software developer and associated with said plurality of software applications, wherein said control file specifies a hierarchy of application classloaders as children of the system classloader, to be used with the modules in said plurality of software applications, and wherein the hierarchy includes a plurality of nested branches, including providing each of said plurality of software applications with its own application classloader hierarchy so that software applications are not aware of classloaders or classes that are assigned to another software application, and wherein the hierarchy is specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications; and

a deployment utility that, upon receiving a request to deploy and run a software application on the server,

parses the control file and determines which classloaders are specified therein for the software application being deployed,

loads with said software application into the Java virtual machine at the server a selection of said application classloaders corresponding to the hierarchy specified by said control file, including, if a particular software application or a module in a software application is being redeployed then loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy, and

enables the server to host multiple isolated software applications within the Java virtual machine, as defined by the hierarchy.

Taylor discloses a method, system, and program for processing objects in a distributed computing environment. A determination is made of a program needed to process a component. A file including the determined program is requested from a remote process. The requested file is received from the remote process, wherein the requested file includes a plurality of sections, each including different programs. At least one section includes programs that are intended only to be executed in a remote address space of the remote process and at least one other section includes programs that are intended to be downloaded from the remote process and execute in a client address space that is different than the remote address space. (Abstract). As further disclosed therein, the population manifest 326 is parsed by the container 306 to construct class loaders as necessary to load the components indicated in the manifests. FIG. 6 illustrates an example of a class loader hierarchy 350 that the population manifest 326 may implicitly define. The

class loader hierarchy 350 divides class loaders into factory, root, and component class loaders. Component class loaders are used to load the variable components of the system, such as services, facility implementations, plug-ins, and dynamically loaded modules. Root class loaders load common interfaces that enable communication between the components of the system, such as facility interfaces. Factory class loaders are used to load factory classes. A factory is a component responsible for instantiating and destroying a particular type (or set of types) of components. A facility is a component whose instances can be shared across other components. (Paragraph [0050]). Each element in the one or more XML files forming the population manifest 326 indicates a component, encapsulated in a CAR file, to be loaded, and the attributes of the element may include information about the component, such as public package, version, etc. During runtime, a new component may be added to the system by adding a new element to the XML file comprising a population manifest 326. A root XML file may maintain information on the root components that are loaded, where the last XML element in the root XML file would comprise the component that is loaded by the lowest root class loader in the hierarchy. (Paragraph [0052]). Alternatively, the remote process, remote address space, and client address space may be implemented in a same computer system, wherein communication with the remote process occurs through a shared memory in the computer system. (Paragraph [0012]).

Susarla discloses providing a dynamic class reloading using a modular, pluggable and maintainable class loader. In one embodiment, to enforce module level separation of the utility classes, the class loader stack may include another layer between the application class loader 202 and the other layers. This layer may load all the classes that are visible only to a module but not cross-module. The table of FIG. 3 illustrates the interdependency between the classes loaded by the class loaders in the class loader stack illustrated in FIG. 4, according to one embodiment. (Column 11, Lines 7-14).

In the Office Action, it was asserted that Taylor discloses providing a plurality of software applications, wherein each of said plurality of software applications includes a plurality of deployable modules and classes associated therewith, and wherein the software applications can be customized by a software developer and then deployed to run on the same server.

However, Applicant respectfully submits that, based on the above description it appears that, in Taylor, the system disclosed therein is used to support programs that are intended only

to be executed in a remote address space of the remote process. Taylor appears to indicate that the remote process, remote address space, and client address space may be implemented in the same computer system, wherein communication with the remote process occurs through a shared memory in the computer system. However, Taylor appears to require the two processes (local and remote) to communicate via shared memory, even when running on the same machine.

Accordingly, Claim 1 has been amended to more clearly define that, in the embodiment therein, the server executes a Java virtual machine that includes a system classloader, and that the deployment utility, upon receiving a request to deploy and run a software application on the server, parses the control file and determines which classloaders are specified therein for the software application being deployed, loads with said software application into the Java virtual machine a selection of the application classloaders corresponding to the hierarchy specified by said control file, and enables the server to host multiple isolated software applications within the Java virtual machine, as defined by the hierarchy. Applicant respectfully submits that neither Taylor nor Susarla, when considered alone or in combination, appear to disclose or render obvious hosting multiple isolated software applications within a Java virtual machine, as defined by Claim 1, as currently amended.

In the Office Action, it was further asserted that Taylor discloses parsing the control file and determining which classloaders are specified therein for the software application being deployed, retrieving a selection of said application classloaders according to the hierarchy specified by said control file, and loading said modules and classes onto the server as part of said software application corresponding to said hierarchy. It was further asserted that Taylor in view of Susarla renders obvious when a particular software application or a module is being redeployed then loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy.

However, Applicant further respectfully submits that, in Taylor, the class loader hierarchy disclosed therein appears to use a manifest file at the application level. Similarly, Figures 7 and 8 appear to show that the class loader hierarchy stops at the population instance for the application, but does not appear to proceed to the module level within each population instance. As such, Applicant respectfully submits that these dynamically loaded modules appear to be

system modules, rather than the modules of an application, since the component classloaders are used to load component modules of the system.

Accordingly, Claim 1 has been amended to more clearly define that, in the embodiment therein, the control file specifies a hierarchy of application classloaders as children of the system classloader, to be used with the modules in said plurality of software applications, and wherein the hierarchy includes a plurality of nested branches, including providing each of said plurality of software applications with its own application classloader hierarchy so that software applications are not aware of classloaders or classes that are assigned to another software application, and wherein the hierarchy is specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications.

In view of the above comments, Applicant respectfully submits that Claim 1, as currently amended, is neither anticipated by, nor obvious in view of, the cited references, and reconsideration thereof is respectfully requested.

Claims 11 and 37

The comments provided above with respect to Claim 1 are hereby incorporated by reference. Claim 37 has been canceled, rendering moot the rejection of this claim. Claim 11 has been amended similarly to Claim 1 to more clearly define the embodiments therein. For similar reasons as provided above with respect to Claim 1, Applicant respectfully submits that Claim 11, as amended, is likewise neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

Claims 2-7, 9, 12-17, 19, 21-27, 29 and 31-34 and 38-42

Claims 38-42 have been canceled, rendering moot the rejection of these claims. Claims 2-7, 9, 12-17, 19 and 31-34 depend from and include all of the features of either Claim 1 or 11. Claims 2-7, 9, 12-17, 19 and 31-34 are not addressed separately, but it is respectfully submitted that these claims are allowable as depending from an allowable independent claim, and further in view of the amendments to the independent claims and the comments provided above. Reconsideration thereof is respectfully requested.

IV. Additional Amendments

Claims 43-46 have been newly added by the present Reply. Applicant respectfully requests that new Claims 43-46 be included in the application, and considered therewith.

V. Request for Interview

In the event the above remarks fail to place the case in condition for allowance, Applicant respectfully requests the opportunity to interview with the Examiner at their convenience, prior to the issuance of a subsequent Office Action, to assist in expediting prosecution. The Examiner is invited to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

VI. Conclusion

In view of the above amendments and remarks, it is respectfully submitted that all of the claims now pending in the subject patent application should be allowable, and reconsideration thereof is respectfully requested. The Examiner is respectfully requested to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

Enclosed is a PETITION FOR EXTENSION OF TIME UNDER 37 C.F.R. §1.136 for extending the time to respond up to and including March 16, 2009. The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 06-1325 for any matter in connection with this Reply, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: March 16, 2009

By: /Karl Kenna
Karl Kenna
Reg. No. 45,445

Customer No.: 23910
FLIESLER MEYER LLP
650 California Street, Fourteenth Floor
San Francisco, California 94108
Telephone: (415) 362-3800